EXPERIMENT NO - 01

Name: Kale Komal Janardan

Div: TE IT – A

Roll No.: ITA539 **Batch: A2**

DOP: **Sign:**

DOS: **Grade:**

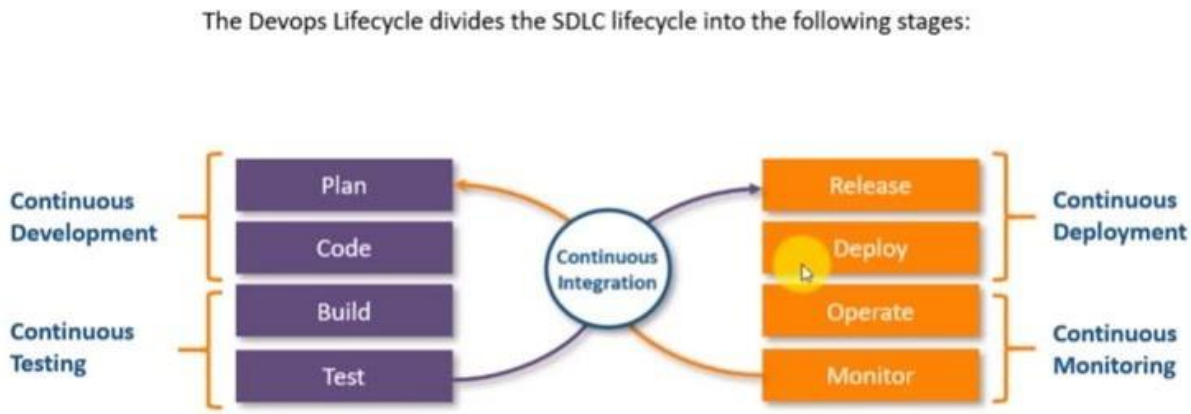
Title: Introduction to DevOps.

Theory:

- **What is DevOps?**

  DevOps is a software development methodology which improves the collaboration between developers and operations team to make software production and deployment in an automated and repeatable way by using various automation tools. These automation tools are implemented using various stages which are a part of the DevOps Lifecycle. The word 'DevOps' is a combination of two words, 'Development' and 'Operations'.

❖ DevOps Lifecycle:

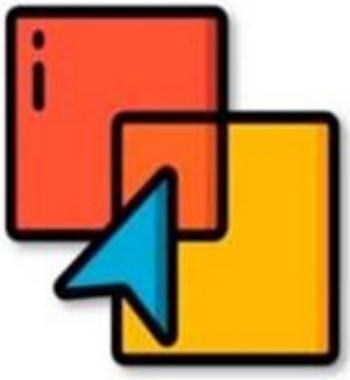


- **Continuous Development:**

  In this DevOps stage the development of software takes place constantly. In this phase, the entire development process is separated into small development cycles. This benefits DevOps team to speed up software development and delivery process. This stage involves committing code to version control tools such as Git or SVN for maintaining the different versions of the code and tools lie Ant, Maven, Gradle for building/packaging the code into an executable file that can be forwarded to the QAs for testing.

- **Continuous Integration:**

In this stage, new functionality is integrated with the prevailing code and testing takes place. Continuous development is only possible due to continuous integration and testing. The stage is critical point in the whole DevOps Lifecycle. It deals with integrating the different stages of the devops lifecycle and is therefore, the key in automating the whole DevOps Process.



- **Continuous Deployment:**

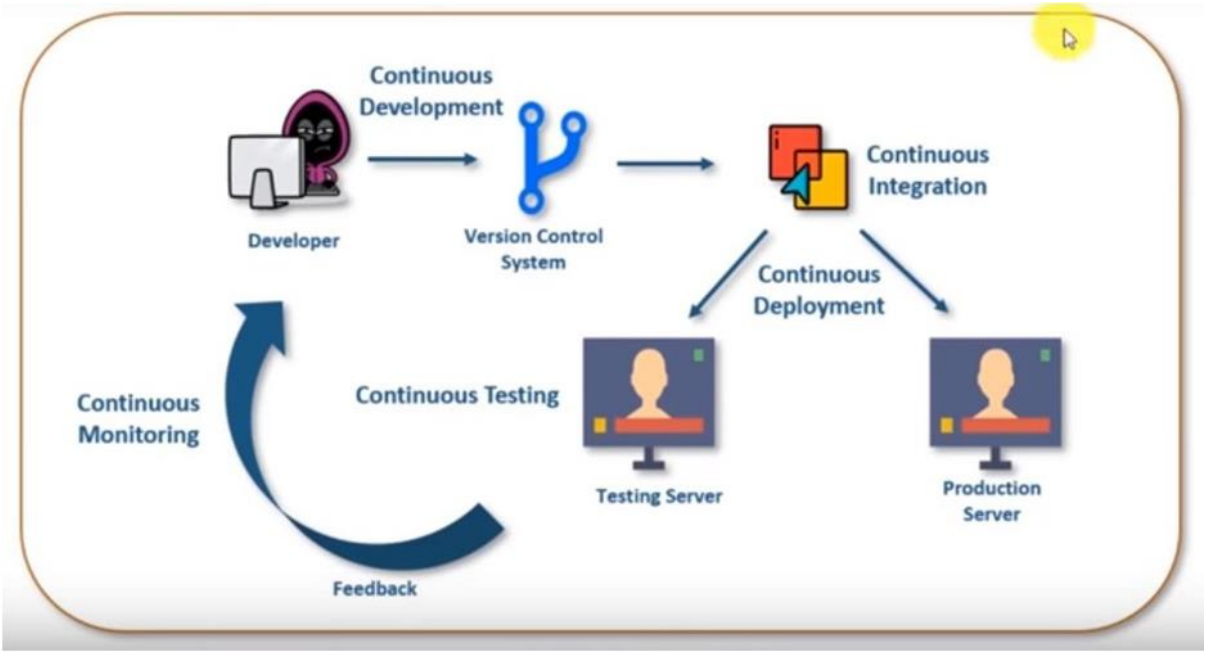
In this phase, the deployment process takes place continuously. It is performed in such a manner that any changes made any time in the code should not affect the functioning of high traffic website. In this stage, the code is built, the environment or the application is containerized and is pushed on to the desired server. The key processes in this stage are Configuration Management, Virtualization and Containerization.

- **Continuous Testing:**

The stage deals with automated testing of the application pushed by the developer. If there is an error, the message is sent back to the integration tool, this tool in turn notifies the developer of the error. If the test was a success, the message is sent to integration tool which pushes the build on the production server.

- **Continuous Monitoring:**

The stage continuously monitors the deployed application for bugs or crashes. It can also be setup to collect user feedback. The collected data is then sent to the developers to improve the application.



❖ **DevOps Tools:**

- **Git:**

Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace. It is used as a critical distributed version-control for the DevOps tool. It is primarily used for source-code management in software development, but it can be used to keep track of changes in any set of files.

- **Jenkins:**

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly.

- **Docker:**

Docker is a high-end DevOps tool that allows building ship and run distributed applications on multiple systems. It also helps to assemble the apps quickly from the components and it is typically suitable for container management. It aims to simplify and accelerate various workflows in your SDLC with an integrated approach.

- **Puppet:**

Puppet is the most widely used DevOps tool. It allows the delivery and release of the technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.

- **Ansible:**

Ansible is a leading DevOps tool. Ansible is an open-source IT engine that automates applications deployment, cloud provisioning, intra service orchestration and other IT tools. It makes it easier for DevOps temas to scale automation and speed up productivity.

- **Selenium:**

Selenium is a portable software-testing framework used for web applications. It is an open source tool which is used for automating the tests carried out on web browsers (Web applications are tested using any web browser).

- **Nagios:**

Nagios is an open-source devops tool which is used for monitoring systems, networks and infrastructure. It also offers monitoring and alerting services for any configurable event. It is one of the most useful tool for Devops.

- **Kubernetes:**

Kubernetes is an open source DevOps tool used to automate deployment and management of containerized applications and perhaps one of the most popular container orchestration tools. It offers own IP addresses and a single DNS name for a set of Pods – Service Delivery and load balancing.

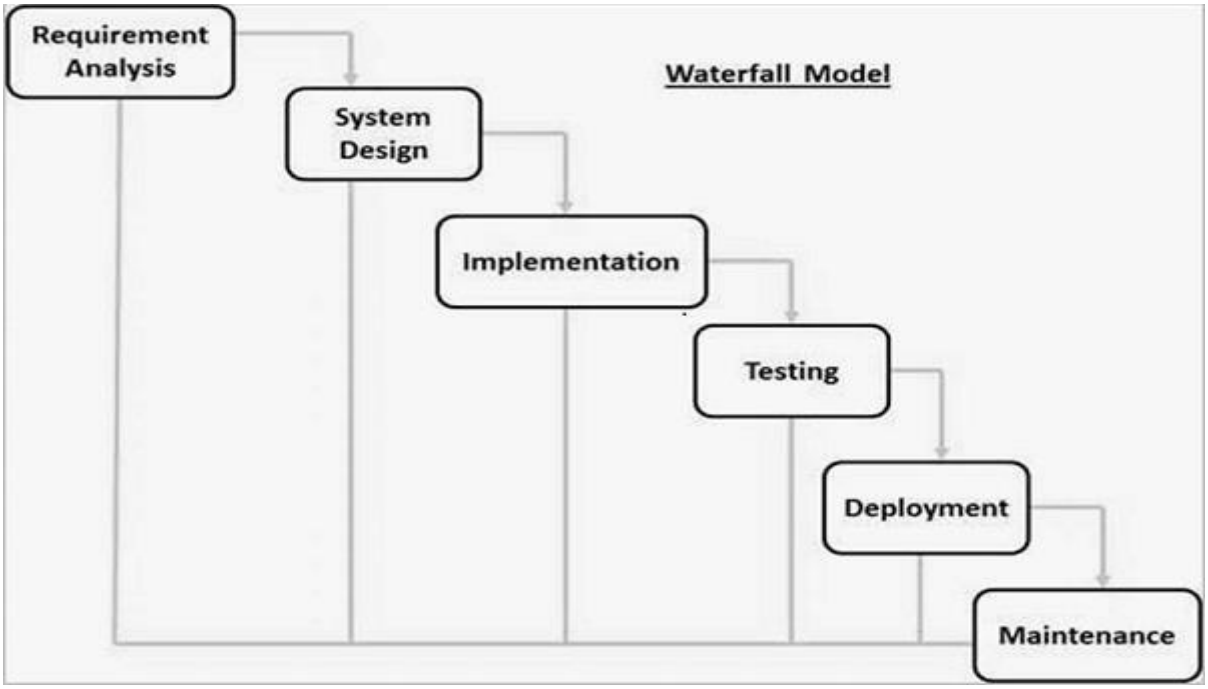
❖ Waterfall Model:

- The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.
- The Waterfall model is the earliest SDLC approach that was used for software development.
- The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

- **Waterfall Model – Design:**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model:



The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** −With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

- **Waterfall Model – Application:**

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

- **Waterfall Model – Advantages:**

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

- **Waterfall Model – Disadvantages:**

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.

- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

❖ **Agile Methodology:**

- Agile Methodology meaning a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.
- An agile methodology is an iterative approach to software development. Each iteration of agile methodology takes a short time interval of 1 to 4 weeks. The agile development process is aligned to deliver the changing business requirement. It distributes the software with faster and fewer changes.
- The single-phase software development takes 6 to 18 months. In single- phase development, all the requirement gathering and risks managementfactors are predicted initially.
- The agile software development process frequently takes the feedback of workable product. The workable product is delivered within 1 to 4 weeks of iteration.

- **Applications Agile Methodology:**

- Adaptive Software Development (ASD)
- Crystal Methods.
- Dynamic Systems Development Method (DSDM)
- Extreme Programming (XP)
- Feature-Driven Development (FDD)
- Lean Software Development (LSD)
- SCRUM.

- **Advantages of Agile Methodology:**

- Customer satisfaction is rapid, continuous development and delivery of useful software.
- Customer, Developer, and Product Owner interact regularly to emphasize rather than processes and tools.
- Product is developed fast and frequently delivered (weeks rather than months.)
- A face-to-face conversation is the best form of communication.
- It continuously gave attention to technical excellence and good design.
- Daily and close cooperation between business people and developers.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed.

- **Disadvantages of Agile methodology:**

- It is not useful for small development projects.
- There is a lack of intensity on necessary designing and documentation.
- It requires an expert project member to take crucial decisions in the meeting.
- Cost of Agile development methodology is slightly more as compared to other development methodology.
- The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.

Conclusion: Studied Introduction to DevOps.